N93-13395

A Compressible Boundary Layer Algorithm for Use with SINDA '85

Barbara Sakowski and Douglas Darling
NASA - Lewis Research Center

Allan van de Wall
Case Western Reserve University

NOMENCLATURE

C_p	specific heat at constant pressure
i	index of grid location in the x-direction
j	index of grid location in the y-direction
jj	total number of grid points in the y-direction
P	pressure
r	radius of wall (axisymmetric case)
Re	Reynolds Number (per unit length)
RUB	coefficient in front of derivatives of the advective terms in the x-direction
RVB	coefficient in front of derivatives of the advective terms in the y-direction
T	temperature
u	velocity component in direction of core flow
v	velocity component perpendicular to core flow
x	location in the direction of the core flow
y	location in the direction perpendicular to the core flow
μ_{eff}	total viscosity (molecular and turbulent)
ρ	density
$_{max}$	core flow value at point with maximum Mach number, used to non-dimensionalize boundary layer equations

INTRODUCTION

At times we need to analyze the thermal behavior of systems that include both conduction and high speed flows. Unfortunately, most high-speed-flow codes have limited conduction capabilities and most conduction codes, such as SINDA, cannot model high speed flows. It would be useful to interface a high-speed-flow solution and SINDA. When interfacing a high-speed-flow solution to SINDA, it may be necessary to include the viscous effects in the energy equations. Boundary layer effects of interest include heat transfer coefficients (including convection and viscous dissipation) and friction coefficients. To meet this need, a fast, uncoupled, compressible, two-dimensional, boundary later algorithm was developed that can model flows with and without separation. This algorithm was used as a subroutine with SINDA. Given the core flow properties and the wall heat flux from SINDA, the boundary layer algorithm returns a

wall temperature to SINDA. SINDA and the boundary layer algorithm are iterated until they predict the same wall temperature.

BOUNDARY LAYER ALGORITHM

Boundary Layer Equations

The forms of the boundary layer equations used in the finite difference scheme were the compressible, parabolized Navier-Stokes (PNS) equations by Roach, et al [1]. The equations were 2-dimensional, viscous, and were solved for the primitive variables. The y-momentum equation simply reduced to

$$\frac{\partial \bar{P}}{\partial y}=0 \tag{1}$$

The other equations were as follows:

continuity

$$\frac{\partial(\bar{\rho}\,\bar{u})}{\partial x}+\frac{\partial(\bar{\rho}\,\bar{v})}{\partial y}=0 \tag{2}$$

x momentum

$$\bar{\rho}\,\bar{u}\frac{\partial \bar{u}}{\partial x}+\bar{\rho}\,\bar{v}\frac{\partial \bar{u}}{\partial y}=\frac{1}{R_o^{\,k}}\frac{\partial}{\partial y}\left(R_o^{\,k}\frac{\partial u}{\partial y}\right)-\frac{\partial P}{\partial x} \tag{3}$$

energy

$$\overline{\rho u}\frac{\partial T}{\partial x}+\overline{\rho v}\frac{\partial \bar{T}}{\partial y}=\frac{1}{R_o^{\,k}}\frac{\partial}{\partial y}\left(\frac{R_o^{\,k}}{RePr}\frac{\partial \bar{T}}{\partial y}\right)-\frac{1}{Re}\left(\frac{\partial \bar{u}}{\partial y}\right)^2+\bar{u}\frac{\partial \bar{P}}{\partial x} \tag{4}$$

The variables were defined as:

$$\bar{\rho}=\frac{\rho}{\rho_{max}} \tag{5}$$

$$\bar{u}=\frac{u}{u_{max}} \tag{6}$$

$$\bar{P}=\frac{P}{\rho_{max}u_{max}^2} \tag{7}$$

$$\bar{T}=\frac{c_p T}{u_{max}^2} \tag{8}$$

$$Re = \frac{\rho_{max} u_{max}}{\mu_{eff}} \tag{9}$$

For axisymmetric flows, (k=1), R_o was defined as:

$$R_o = r \pm y\cos\theta \tag{10}$$

where r was the radius of the wall, θ was the angle of the wall, and the plus or minus signs referred to external or internal flows respectively. For rectangular coordinates R_o was ignored, (k=0). The coordinates x and y used in the above equations were the transformed x and y coordinates, that is, the x coordinate followed along the wall surface and the y coordinate was perpendicular to the wall. It should be noted that Re was not actually non-dimensional. Re had units of $(\text{length})^{-1}$. Also, x and y (and R_o) remained dimensional. So, each term in each of the above boundary layer equations had units of $(\text{length})^{-1}$.

Solution Algorithm

The momentum, continuity and energy equations were differenced as described by Kwon et al.[2] For any scalar quantity ϕ :

$$\left(\frac{\partial\phi}{\partial x}\right)_{i,j} = \frac{\phi_{i,j}-\phi_{i-1,j}}{x_i - x_{i-1}} \tag{11}$$

$$\left(\frac{\partial\phi}{\partial y}\right)_{i,j} = \frac{\phi_{i,j+1}-\phi_{i,j-1}}{y_{j+1} - y_{j-1}} \tag{12}$$

$$\left(\frac{\partial}{\partial y}\frac{1}{Re}\frac{\partial\phi}{\partial y}\right)_{i,j} = \frac{\dfrac{1}{Re_{i,j+\frac{1}{2}}}\dfrac{\phi_{i,j+1}-\phi_{i,j}}{y_{j+1}-y_j} - \dfrac{1}{Re_{i,j-\frac{1}{2}}}\dfrac{\phi_{i,j}-\phi_{i,j-1}}{y_j - y_{j-1}}}{\frac{1}{2}(y_{j+1}-y_{j-1})} \tag{13}$$

Due to the parabolic nature of the boundary layer equations, the governing

equations were solved by marching from station to station in the direction of the core flow. Solutions at a given station were obtained by solving the boundary layer equations sequentially. First, the momentum equation was solved for the velocity component in the core flow (streamwise) direction (u). Second, continuity was solved for the velocity component in the y direction (v). Then, the energy equation was solved for the temperature (T). If the station had not converged, the momentum, continuity, and energy equations were solved again for the velocity components and temperature, using the flow properties from the previous iteration. Convergence at a given station was obtained when the streamwise velocity components at all grid locations at that station converged. After the calculation at the station converged, the algorithm marched to the next streamwise station to solve for the boundary layer properties. This streamwise marching continued throughout the entire domain.

Grid spacing perpendicular to the wall (y-direction) was based on an exponential function. The grid spacing was fine near the wall to better resolve the gradients at the wall. The grid spacing was course away from the wall where fine resolution of the gradients normal to the core stream were not necessary.

The derivative of the velocity at the wall (used to calculate wall shear stress) was determined using a second order approximation. Values at j=1 (wall), j=2, and j=3 were used.

$$\left(\frac{\partial u}{\partial y}\right)_{i,1} = \frac{y_{i,3}^2(u_{i,2}-u_{i,1}) - y_{i,2}^2(u_{i,3}-u_{i,1})}{y_{i,2}y_{i,3}(y_{i,3}-y_{i,2})} \tag{14}$$

The above algorithm worked well for flows with weak viscous/inviscid interaction, since the downstream influence could be neglected. However, when the flows were strongly interacting, such as those with strongly adverse pressure gradients or separation, downstream conditions had to be considered. To account for downstream influences the pressure gradient was differenced as a weighted average of forward and backward differences. This techniques was based on the method of Davis and Barnett[3].

Specifically the pressure gradient was differenced as

$$\frac{\partial \overline{P}}{\partial x} = (\epsilon)\left(\frac{\overline{P_{i,j}} - \overline{P_{i-1,j}}}{x_i - x_{i-1}}\right) + (1-\epsilon)\left(\frac{P_{i+1,j} - P_{i,j}}{x_{i+1} - x_i}\right) \tag{15}$$

where ϵ is a weighting parameter of the forward and backward differencing. This term was required to remove the ellipticity in the PNS equations in strongly interacting flows. The quantity ϵ determined what fraction of the forward difference of the pressure

gradient can be included so that the equations remain non-elliptic. If the flow was supersonic at a given j location, then only backward differencing was used for the pressure gradient ($\epsilon = 1$). If the flow was subsonic then at a given j location then the following expression for ϵ was used,

$$\epsilon = \frac{\gamma M^2}{1 + (\gamma - 1) M^2} \tag{16}$$

Boundary Layer Separation

Flaring of the advective term parallel to the core flow was used when the flow was separated. This was handled by taking the absolute value of the coefficient in front of the derivative of the advective term (RUB) and multiplying it by .1 to make it smaller. Again this operation was only performed when the flow was separated.

Boundary Layer Turbulence Model

A modified Baldwin-Lomax model was used to account for turbulence. This model was a zero equation, eddy viscosity model. This model was faster than other turbulence models, such as the k-ϵ two equation model. A modified model of Visbal and Knight was used to handle the effects of separation. However, the model of Visbal and Knight required further modification, the most important of which was the modification of the Baldwin-Lomax parameter C_{cp}. C_{cp} needed to be a function of both core flow Mach number and core flow pressure gradient. This modification was extremely important in matching numerical results and experimental data. A detailed description of the turbulence model used is given in Sakowski, et. al [4].

Boundary Layer/Core Flow Interface

The boundary layer algorithm serves as a link between the conduction program (SINDA) and an inviscid core flow program. The interface with SINDA will be discussed in a later section. In this section we will look at the interface of the boundary layer algorithm with a core flow program.

When the boundary layer algorithm was interfaced with a core flow algorithm, it was necessary that the boundary layer properties smoothly approached the core flow values. That is, when the derivatives perpendicular to the wall were zero, the core flow values had to be a solution to the boundary layer equations. The complicated part was matching the boundary layer and core flow, since the boundary layer and core flow algorithms were probably not differenced in the same way. If the differential equations were solved exactly there would not be problem, but they were not solved exactly. What the core flow algorithm predicted as a solution, was not exactly what the boundary layer algorithm predicted as a solution as the y-derivatives went to zero (far from the wall).

The difference was usually fairly small (2% or so), but this small difference could have a big effect on the integral performed to calculate the displacement thickness. An adjustment in the way the pressure gradient term was calculated in the boundary layer algorithm forced the boundary layer properties to smoothly approach the core values. Without this adjustment, the displacement thickness, predicted by the boundary layer program had large errors. The adjustment of the pressure gradient was performed by solving the finite differenced momentum equations for dP/dx when all y-derivatives were zero.

$$\frac{\partial \bar{P}}{\partial x}=-\bar{\rho}_e\,\bar{u}_e\frac{\partial u_e}{\partial x} \tag{17}$$

The corrected value of the pressure derivatives was calculated with the edge values from the core flow algorithm using the same differencing scheme used in the boundary layer algorithm. In this way the boundary layer algorithm approached the core flow values as the y-derivatives approached zero.

Another consideration for interfacing a core flow algorithm and the boundary layer code was stability. At times the boundary layer algorithm has a stability problem. This problem tended to initiate near the edge of the boundary layer. From one iteration to the next the values near the core flow sometimes fluctuated between less than and greater than the core flow value. Sometimes these fluctuations died out and the program converged. However, other times the oscillations grew, causing the calculations to diverge. To solve this problem flaring was used. RVB was part of the advective terms in the y-direction. RVB was the coefficient in front of the $\partial u/\partial y$ term in the finite differenced x-momentum equation, and the $\partial T/\partial y$ term in the finite differenced energy equation. These were the advective terms in the y-direction. Without flaring RVB was simply ρv. With flaring RVB was changed as follows:

$$RVB=K_1|\,\bar{\rho}\,\bar{v}\,| \tag{18}$$

K_1's for momentum equation

	if $u<u_e$	if $u>u_e$
if $\frac{\partial u}{\partial y}>0$	$K_1=-\frac{u}{u_e}$	$K_1=\left(1-\frac{u}{u_e}\right)$
if $\frac{\partial u}{\partial y}<0$	$K_1=\left(1-\frac{u}{u_e}\right)$	$K_1=-\frac{u}{u_e}$

K_1's for energy equation

$$
\begin{array}{lcc}
 & \text{if } T<T_e & \text{if } T>T_e \\
\text{if } \dfrac{\partial T}{\partial y}>0 & K_1=-\dfrac{T}{T_e} & K_1=\left(1-\dfrac{T}{T_e}\right) \\
\text{if } \dfrac{\partial T}{\partial y}<0 & K_1=\left(1-\dfrac{T}{T_e}\right) & K_1=-\dfrac{T}{T_e}
\end{array}
$$

The motivation for the above flaring was to make the core flow value a numerically stable solution in the boundary layer algorithm far from the wall. This flaring was found to be very important to help the stability of the algorithm, particularly when there was an adverse pressure gradient, separation, bleeds, or bypasses. For more detail on interfacing the boundary layer algorithm of Roach, et al. with a core flow, refer to Darling, et al. [5].

CORE FLOW INPUT TO BOUNDARY LAYER ALGORITHM

It was mentioned previously that the boundary layer algorithm required as an input, a core flow. The inviscid core flow variables needed by the boundary layer algorithm were the Mach number, temperature, and pressure. For flows where the interaction between the core flow and boundary layer were negligible, the interface was very simple and easily implemented. All that was required was an input file by the name **LBL.DAT**. This file contains seven namelists described as follows:

NAMELIST/GEOM/X:	**Grid point locations that run parallel to the centerline of the wall**
NAMELIST/AREAX/AREA:	**Surface area of the wall that corresponds to a respective X grid point location**
NAMELIST/TEMP/T1:	**Core flow temperature that corresponds to a respective X grid point location**
NAMELIST/PRES/P1:	**Core flow that corresponds to a respective X grid point location**

NAMELIST/MACH/AM: **Core flow Mach number that corresponds to a respective X grid point location**

NAMELIST/RADIUS/RADY: **Radius of the wall measured from the wall centerline and perpendicular to its respective X grid point location**

NAMELIST/BL/

JJJ: **Number of grid points perpendicular to the wall**

DUMX: **Convergence criteria on the u velocity component of the x momentum equation**

II: **Number of grid points in the streamwise direction**

TURB: **Flag to signal use of the Baldwin-Lomax turbulence model:**
TURB=.TRUE. - turbulence on
TURB=.FALSE. - turbulence off

ENGU: **Flag to signal use of units:**

ENGU	T1	P1	X	AREA
ENG UNITS - .TRUE.	°R	lbf/ft^2	ft	ft^2
SI UNITS - .FALSE.	°K	N/m^2	m	m^2

AXI: **Flag to signal axisymmetric flow:**
AXI=.TRUE. - axisymmetric flow
AXI=.FALSE. - 2-D flow

EXT: **Flag to signal external flow:**
EXT=.TRUE. - external flow
EXT=.FALSE. - internal flow

MYES: **Flag to print to:**
Mach.out: Prints non-dimensional velocity, temperature, density, and pressure profiles for each x grid location and for the first 10 SINDA iterations - after 10 SINDA iterations, the profiles are printed every 20 SINDA iterations

Each x grid location also gives the following parameters:

IT: number of boundary layer iterations

M: core flow mach number

Cf: skin friction coefficient

Ybl: distance of the last y grid point from the wall

C2.OUT: **Prints the following boundary layer parameters:**

IT: number of boundary layer iterations

Th: momentum thickness

Cf: skin friction coefficient

Disp: displacement thickness

for the first 10 SINDA iterations - after 10 SINDA iterations, the profiles are printed every 20 SINDA iterations

A sample LBL.dat file is provided in the APPENDIX.

For flows where there was a strong interaction between the core flow and boundary layer, the simple input data file of constant inviscid core flow values would change complexion. The file would become a core flow algorithm which continuously updates the core flow variables to account for the boundary layer interaction, such as flows with shock waves. Such algorithms are not discussed in this paper.

SINDA/BOUNDARY LAYER INTERFACE

The nodes in SINDA that represent the wall surface nodes MUST be declared as boundary nodes. This is done to obtain heat rates on the nodes that can be sent to the boundary layer algorithm. For a steady state solution the heat rate on the surface node will be zero if the nodes are defined as arithmetic or diffusion nodes. This is because SINDA effectively "sees" an insulated surface. In actuality the surface is not insulated, because of the presence of the boundary layer flowing over it. For each SINDA iteration, the heat rates from the SINDA nodes are passed to the boundary layer where new wall temperatures are determined. The boundary layer algorithm will continue iterating until it has a converged solution. After the boundary layer algorithm has converged, the boundary layer wall temperatures become the new SINDA boundary node temperatures. Thus the SINDA boundary node temperatures are updated every SINDA

iteration. The program iterates between SINDA and the boundary layer algorithm until the boundary layer wall temperatures match the SINDA boundary node temperatures.

SINDA INPUT TO BOUNDARY LAYER ALGORITHM

In **HEADER CARRAY DATA**, reserve the following variables:

1=LBL.DAT
2=MACH.OUT
3=C2.OUT

Also, the common SINDA/FORTRAN variable **BTEST** should not be used. In **HEADER VARIABLES** 1, make the call to the subroutine **INTERFACE** as follows:

CALL INTERFACE(A,B,C,D,E,ABSZRO,BTEST,UCA1,UCA2,UCA3)

where **A** = Submodel name in quotes where the surface boundary nodes are located, and no more than 8 characters long.

B = SINDA node number of the surface boundary node located at the last x grid location of the boundary layer (integer).

C = SINDA node number of the surface boundary node located at the first x grid location of the boundary layer (integer).

D = Number by which the SINDA boundary nodes are incremented (integer).

E = Units used by SINDA. 'ENG' for English units. 'SI' for SI units.

The remaining arguments should be left as they are. ABSZRO is the SINDA variable for absolute temperature defined in the HEADER OPTIONS DATA BLOCK. BTEST is a counter for the number of SINDA iterations. UCA1, UCA2, and UCA3 are the SINDA variables for the input and output file names defined in the HEADER CARRAY DATA BLOCK.

CONCLUSIONS

A fast steady, compressible, turbulent boundary layer algorithm that can be used to model separated flows has been written as a subroutine for SINDA. Results from the boundary layer algorithm compared well with experimental pressure distributions when the boundary layer was interactive with the core flow, Darling, et al. [5] and Roach, et al

[1]. In addition, the friction coefficients and momentum thicknesses predicted by the boundary layer code compared well with experimental data, Roach, et al. [1]. The Baldwin-Lomax turbulence model was used following the modifications of Sakowski, et al [4], also matched experimental data fairly well. Currently comparisons are being made with experimental data validate the heat transfer predictions of the boundary layer algorithm. The boundary layer algorithm was found to converge quickly with SINDA. A simple SINDA model with 25 x-grid locations was tested, see the APPENDIX for the SINDA input. The model converged in 95 SINDA iterations.

REFERENCES

1. R.L. Roach, C. Nelson, B.A. Sakowski, D.D. Darling, A.G. van de Wall, "A Fast, Uncoupled, Compressible, Two-Dimensional, Unsteady Boundary Layer Algorithm with Separation for Engine Inlets,' AIAA 92-3082, 1992.

2. O.K. Kwon, R.H. Pletcher, R.A. Delaney, "Solution Procedure for Unsteady Two-Dimensional Boundary Layers," ASME Journal of Fluids Engineering, Vol. 110, March 1988, pp 69-75.

3. R.T. Davis, M. Barnett, "The Calculation of Supersonic Viscous Flows using the Parabolized Navier-Stokes Equations," Computers and Fluids, Vol. 14, No. 3, pp 197-224, 1986.

4. B.A. Sakowski, D.D. Darling, R.L. Roach, A.G. van de Wall, "Evaluation and Application of the Baldwin-Lomax Turbulence Model in Two-Dimensional, Unsteady, Compressible Boundary Layers with and without Separation in Engine Inlets," AIAA 3676, 1992.

5. D.D. Darling, B.A. Sakowski, "Interface of an Uncoupled Boundary Layer Algorithm With an Inviscid Core Flow Algorithm for Unsteady Supersonic Engine Inlets," AIAA 92-3083, 1992.

APPENDIX

```
&GEOM X=.225528E-2,.247675E-1,.515367E-1,.778460E-1,.107633,.128037,.151619,
        .174448,.203611,.228309,.231856,.235408,.238954,.242498,
        .246048,.249659,
        .25352,.258018,.262899,.267638,.272238,.276733,
        .281173,.285603,.290036,.294476,.298924,.303379,.307842,.312309,
        .316781,.321258,.325737,.330218,
        .334703,.343673,.352644,.366070           &END
&AREAX AREA=1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
            1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
            1.0,1.0,1.0,1.0,1.0,1.0 &END
&TEMP T1=1982.9,1983.3,1983.5,1983.6,1983.5,1983.3,1982.7,1981.9,1980.3,
         1975.5,1973.5,1970.8,1966.,1960.4,1945.2,1917.8,1857.5,1776.7,
         1707.9,1605.9,1526.4,1468.8,1385.8,
         1325.1,1283.,1253.3,1228.5,1207.9,
         1188.9,1171.,1155.3,1142.7,1131.3,1122.4,
         1116.5,1106.3,1101.8,1098.6          &END
&PRES P1=.172115E7,.172253E7,.17236E7,.172399E7,.172346E7,.172235E7,
         .17199E7,.171615E7,.170832E7,.168388E7,.167384E7,.165993E7,.163667E7,
         .160348E7,.153294E7,.141769E7,.122549E7,
         .101028E7,.810816E6,.613865E6,.492139E6,.407487E6,
         .315702E6,.257376E6,.222281E6,.20018E6,.183118E6,.169834E6,
         .158537E6,.148288E6,.139641E6,.133107E6,.127322E6,.122975E6,
         .120144E6,.115385E6,
         .113278E6,.111714E6                  &END
&MACH AM=.389E-1,.393E-1,.411E-1,.442E-1,.492E-1,.541E-1,.619E-1,.73E-1,
         .105,.192,.217,.25,.29,.354,.442,.561,.725,.919,1.13,1.3,1.48,1.6,
          1.77,1.89,1.98,2.01,2.1,2.14,2.19,2.22,2.25,2.29,2.31,2.32,2.34,
          2.36,2.37,2.374                                               &END
&RADIUSY RADY=1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
            1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
            1.0,1.0,1.0,1.0,1.0,1.0 &END
&BL    JJJ=100,DUMX1=.001,II=25,TURB=.TRUE.,
       ENGU=.FALSE.,AXI=.TRUE.,EXT=.TRUE.,MYES=.TRUE.   &END
```

```
HEADER OPTIONS DATA
TITLE BOUNDARY LAYER CODE INTERFACE WITH SINDA
      MODEL  = BL1
      OUTPUT = BL2.OUT
      USER1  = BL1.USR
      USER2  = BL2.USR
HEADER CONTROL DATA, GLOBAL
      NLOOPS = 4000
      ABSZRO = -460.0
      UID    = ENG
      ARLXCA = .01      $DEFAULT VALUE
      DRLXCA = .01      $DEFAULT VALUE
      EBALSA = .01      $DEFAULT VALUE
HEADER USER DATA, GLOBAL
C
HEADER CARRAY DATA, BL
       1=NASA$PFSD:[AMBER.SINDA]LBL.DAT
       2=TDISK$DIR:[AMBER]MACH.OUT
       3=TDISK$DIR:[AMBER]C2.OUT

HEADER USER DATA, BL
C
C>>>>>>RESERVE BTEST FOR USE IN BOUNDARY LAYER
HEADER ARRAY DATA, BL
C

       1=  81.,0.1139      $ SPECIFIC HEAT VS. TEMPERATURE
          261.,0.1230      $ UNITS:  BTU/LBM/DEG. F
          621.,0.1330      $ AISI 304 S.S.
          981.,0.1390
         1341.,0.1459

       2=  81., 8.61       $ THERMAL CONDUCTIVITY VS. TEMPERATURE
          261., 9.59       $ UNITS:  BTU/HR/FT/DEG. F
          621.,11.44       $ AISI 304 S.S.
          981.,13.06
         1341.,14.68

C
HEADER NODE DATA, BL

C******AISI 304 S.S./DENSITY=493 LBM/FT**3
C******GENERATE 25 DIFFUSION NODES TO REPRESENT THE WALL!!

       SIM 8801,25,1,70.,A1,5.

C******GENERATE SURFACE NODES!!!!!!!!>>>CONNECT TO THE BOUNDARY LAYER CODE

       GEN -1,25,1,3000.,0.0

C******GENERATE OTHER BOUNDARY NODES TO SIMULATE THE EFFECT OF A SIMPLE
C******ACTIVE COOLING SYSTEM WHOSE EFFECTIVE TEMPERATURE IS 100 DEG F!!!

       GEN -1101,25,1,100.0,0.0

HEADER CONDUCTOR DATA, BL

C********CREATE CONDUCTORS IN THE WALL ALONG THE "X AXIS"
C********FOR THIS CASE THE "X AXIS" FOLLOWS THE DIRECTION OF THE CORE FLOW
         SIM 801,24,1,8801,1,8802,1,A2,6.
```

```
C*********CREATE CONDUCTORS THAT CONNECT THE WALL DIFFUSION NODES
C*********TO THE BOUNDARY NODES WHICH INTERFACE WITH THE BOUNDARY LAYER
        SIM 8801,25,1,8801,1,1,1,A2,.1

C*********CREATE CONDUCTORS THAT CONNECT THE WALL DIFFUSION NODES
C*********TO THE BOUNDARY NODES THAT SIMULATE ACTIVE COOLING
        SIM 88801,25,1,8801,1,1101,1,A2,.1

HEADER VARIABLES 1, BL

C*****THIS IS THE SUBROUTINE THE DOES IT ALL!!!!!!!
      CALL INTERFACE('BL       ',25,1,1,'ENG',ABSZRO,
     &   BTEST,UCA1,UCA2,UCA3,)

HEADER OPERATIONS DATA
C
BUILD BL1,BL
C
       CALL STDSTL

    •

    •

    •

    •

    •
```